ARM in Space

Patrick H. Stakem

Table of Contents

Author

Mr. Patrick H. Stakem received a Bachelors degree in Electrical Engineering from Carnegie-Mellon University, and Masters Degrees in Physics and Computer Science from the Johns Hopkins University.

He began his career in Aerospace with Fairchild Industries on the ATS-6 (Applications Technology Satellite-6), program, a communication satellite that developed much of the technology for the TDRSS (Tracking and Data Relay Satellite System). At Fairchild, Mr. Stakem made the amazing discovery that computers were put onboard the spacecraft. He quickly made himself the expert on their support. He followed the ATS-6 Program through its operation phase, and worked on other projects at NASA's Goddard Space Flight Center including the Hubble Space Telescope, the International Ultraviolet Explorer (IUE), the Solar Maximum Mission (SMM), some of the Landsat missions, and others. He was posted to NASA's Jet Propulsion Laboratory for the MARS-Jupiter-Saturn (MJS-77), which later became the Voyager mission, which is still operating and returning data from outside the solar system at this writing. He has received NASA's Space Shuttle Program Managers commendation award.

Mr. Stakem is affiliated with the Whiting School of Engineering of the Johns Hopkins University. He has direct hands-on experience with the ARM family, linux, and Beowulf.

By 2010, over 6 billion ARM chips had been sold, mostly into the smartphone market. ARM is the target architecture for the GNU/linux-based Android operating system, and the ARM has ports of OpenSolaris, FreeBSD, OpenBSD, NetBSD, various GNU/linux variations, including Gentoo, Debian, Slackware, and Ubuntu, and Windows Embedded.

Introduction

The ARM processor has come a long way from an obscure British microprocessor of the 1980's to being the dominant basis for the current generation of smart phones, and tablet computers, as well as its use in space They are also in television set-top boxes, routers, and embedded applications. The ARM architecture units represent the highest volume of 32-bit processors being shipped, as of this writing.

The ARM started out on the desktop, but along the way, got diverted mostly to embedded use. Now, more powerful ARM chips are challenging the pc and server markets.

ARM is the baseline for small computers and embedded controllers such as the RaspberryPi and the Arduino series. These are provided as licensable products by ARM Holdings Ltd. ARM Holdings is owned by Softbank Group's Vision Fund. Softbank is a Japanese multinational holding company. ARM Holdings does not produce chips. The Intellectual Property (IP) of the ARM is a proprietary, license-able product of ARM Holdings. Numerous company's produce a wide variety of ARM chips, under the license.

ARM's can be found on space missions, most likely in Cubesats, but also on the ISS. At the moment there are several models of a rad-hard Arduino-class microcontroller, and an ongoing NASA-USAF project to produce a radiation hard version of an Arm, a number cruncher capable of hosting an operating system.

One of the big problems in space electronics is the environment, particularly that of radiation. There are various mitigation's, and ARM's have flow on satellite missions, some of which are discussed here

The Surrey Satellite Technology Nanosat Applications Platform

(SNAP-1) was launched on June 28, 2000. The onboard computer (OBC) was based on Intel's StrongArm SA-1100 with 4 Mbytes of 32-bit wide EDAC protected SRAM. The error correction logic could correct 2 bits in every 8 using a modified Hamming code and the errors were "washed" from memory by software to prevent accumulation from multiple single-event upsets. There was 2 Mbytes of Flash memory containing a simple bootloader which loaded the application software into SRAM.

The NanoMind A712D is an onboard computer for Cubesats. It uses as 32-bit ARM cpu, with 2 megabytes of RAM, and 8 megabytes of flash memory. It can also support a MicroSD flash card. It has a Can bus and a I^2C interface. It comes with an extensive software library and real time operating system. Special applications, such as attitude determination and control code are available. It is tolerant to temperatures from -40 to 85 degrees C, but is not completely rad-hard.

The CFC-300 from InnoFlight Inc. of San Diego is another example. It uses the Xilinx Zynq System-on-a-chip architecture. That provides both FPGA capability, and an Arm Cortex A-9 dual core cpu. It has 256 Megabytes of SDRAM, and 32 megabytes of flash. There are multiple synchronous serial interfaces. Daughter cards provide support for SpaceWire, Ethernet, RapidIO, RS-422, and thermistor inputs and heater drive outputs. It can be used with linux or VxWorks.

The Intrepid Cubesat OBC from Tyvak Uses a 400 MHz Atmel processor, and has 128 Mbytes of SDRAM, and 512 Mbytes of flash memory. It draws between 200-300 milliwatts. It includes a command and data handling system, and an onboard electrical power controller. It supports ethernet, RS-232, USB, and the SPI and I^2C interfaces. It includes a JTAG debugging interface. Similar to the Arduino, it supports 3-axis gyros, a 3-axis magnetometer, accelerometers, and a variety of i^2c-interfaced sensors. The Microcontroller is an ARM architecture, with digital signal processing extensions. It has a built-in Image Sensor interface.

There are now several Arduino-class microcontrollers available in rad-hard versions, and one ARM Raspberry-Pi like product being developed by Boeing for NASA and the Air Force with delivery in 2021.

ARM Cortex

We will briefly discuss the ARM architecture families. The ARM Cortex processors are the latest in the 32-bit series, and extend into multicore and 64-bit models for higher performance. There are three basic models of the Cortex processors, targeting different applications areas. These are the Cortex-A, Cortex-M, and Cortex-R. The Cortex models began to appear in 2004.

Cortex-A

Cortex-A processors are targeted to the smartphone and mobile computing markets, as well as digital television, set-top boxes, and printers. They include a memory management unit, a prerequisite to supporting modern operating systems.

The Cortex-A5 optionally supports floating point and NEON media processing. The memory management was improved, and virtual memory is supported. The design is targeted to energy efficiency. Voltages are 1.0 or 1.1 volts, with clock speeds to 1 GHz. Up to 4 cores are supported. Instruction and data L1 caches can extend to 64k each.

NEON implements an advanced SIMD instruction set and was first introduced with the ARM Cortex –A8 model. This is an extension of the FPU with a quad Multiply-and-Accumulate (MAC) unit and additional 64-bit and 128-bit registers.

Single instruction, multiple data (SIMD) describes computers with multiple processing elements that perform the same operation on multiple data simultaneously. Thus, such machines exploit data

level parallelism. Vector processing is where a single operation is applied to all elements of an array of data.

The Cortex-A8 is a superscalar architecture, with dual instruction issue. The NEON SIMD unit is optional, as is floating point support. The architecture supports Thumb-2 and Jazelle, but only single core. Advanced branch prediction algorithms provide an accuracy rate reportedly approaching 95%. Up to a four megabyte Level-2 cache is provided. There is a 128-bit SIMD engine. Cortex A-8 chips have been implemented by Samsung, TI, and Freescale, among others.

The Cortex-A9 can have multiple cores that are multi-issue superscalar and support out-of-order and speculative execution using register renaming. It uses an 8-stage pipeline. Two instructions per cycle can be decoded. There are up to 64k of 4-way set associative Level-1 cache, with up to 512k of Level 2. A 64-bit Harvard architecture memory access allows for maximum bandwidth. Four doubleword writes take five machine cycles. Floating point units and a media processing engine are available for each core. The Cortex-A supports the Thumb-2 instructions, which are 16-bit subsets of ARM instructions.

The Cortex-A9 is implemented in a series of system-on-a-chip devices from multiple manufacturers. As an example, the STMElectronics SPEAr1310 is a dual-core Cortex-A SMP. It has dual cores, and can support either symmetric or asymmetric multiprocessing. It has a 32k instruction and a 32k data cache at Level 1. The Level 2 cache is unified, and is 512k bytes in size. The on-chip inter-processor bus is 64 bits wide. The chip includes 32k of bootrom and 32k of SRAM, with support for external NAND or NOR flash and static ram.

The Cortex chip includes dual Gigabit ports, three fast Ethernet, three PCIe links, three SATA ports, dual USB-2, dual CAN bus, dual HDLC controllers, dual I^2C ports, and six UARTs operating

up to 5 Megabaud. There is an integrated LCD controller with touchscreen support, a keyboard controller, and a memory card interface. It supports thirteen timers and a real time clock, in addition to a cryptographic accelerator. Included are dual 8-channel DMA controllers, a JPEG codec in hardware, and 8-input, 10-bit A/D, and JTAG support.

The Cortex-A15 is multicore, and has an out-of-order superscalar pipeline. The chip was introduced in 2012, and is available from TI, NVidia, Samsung, and others. It can address a terabyte of memory. The integer pipeline is 15 stages long, and the floating point pipeline is up to 25 stages. The instruction issue is speculative. There can be 4 cores per cluster, two clusters per chip. Each core has separate 32k data and instruction caches. The level-2 cache controller supports up to 4 megabytes per cluster. DSP and NEON SIMD are supported, as is floating point. Hardware virtualization support is included. Both Thumb-2 and Jazelle modes are included.

Hardware assisted virtualization is an example of platform virtualization. It uses assistance from the hardware to provide full virtualization, so unmodified guest operating systems can be supported. The technique was first used on IBM System/370 mainframe machines in 1972.

Virtualization is done with a second stage of address translation with its own page tables. I/O can be virtualized. The Hypervisor runs in a new privilege mode, unique to it. The mode is entered with a Hypervisor Call. The Virtualization privilege mode is a new third privilege level. There is the user code level, the operating system level, the Hypervisor level, and a TrustZone Privilege level, at the top. The Embedded Xen product also supports virtualization on the ARM architecture.

In the ARM, before virtualization, the Operating System controlled the memory resource. There is now a second level of address translation. Where virtual addresses used to map to physical

addresses, they now map to Intermediate addresses, which are then mapped to physical addresses by the Hypervisor.

Interrupts are another issue. An interrupt might need to go to the current or another guest operating system, the Hypervisor, or an operating system in the TrustZone. Physical interrupts go to the Hypervisor first; if they need to go to a guest operating system, this is handled by a “virtual” interrupt.

Since the ARM architecture uses memory-mapped I/O, that process is also virtualized. Virtual devices are created by emulation.

Cortex-R

The ARM Cortex-R product addresses hard real-time, safety critical applications. It has specific features to address performance in real-time applications. These include an instruction cache and a data cache, a floating point coprocessor, and an extended 8-stage pipeline. Cortex-R supports the Thumb and Thumb-2 instructions as well as ARM. Up to 64-bit data structures are supported. The compiler must be aware of which architecture is used as the code target, to introduce the proper optimizations for the various models. As with different implementations of the ISA-32 instruction set from Intel, different implementation architectures require different optimization strategies. The correct optimization for one chip could be the worst-case approach for a different implementation. Cortex-R addresses applications in robotics and avionics/space applications.

Cortex-M

The Cortex M models are microcontrollers. There are four models, the Cortex-M0, 1, 2, and 3. All are binary compatible. The M0 and M1 are based on the ARMv6, the M3 is based on the ARMv7, and the M4 is based on the ARM-V7-ME. The Thumb and Thumb-2

subsets are supported. The M3 model has a single cycle 32x32 hardware multiply and 10-12 cycle hardware divide instruction. The M4 adds Digital Signal Processing instructions such as a single-cycle 16/32 bit multiply-accumulate, and supports full Thumb and Thumb-2 instruction set. The IEEE-754 floating point unit is included with the M4. A nested, vectored interrupt controller is included. The 256 interrupts are fully deterministic, and an NMI is included. Cortex-M does not support the instruction and data caches, or the coprocessor interface, and has only a 3-stage pipeline. Only the M3 and M4 models support the Memory Protection Unit. The M3 instruction set provides a pair of synchronization primitives for a guaranteed atomic read-modify-write operation, which is critical to real-time operating systems.

Examples of the chip include the Atmel SAM3 series and the TI Stellaris models. These include flash and sram, timers including a real-time clock and watchdog, PWM for motor control, Ethernet, CAN, USB, and UART functions, and A/D's. The M4 models are made by Atmel, Freescale, STMicroelectronics, and TI.

The SAM RH71 microcontroller product from Microchip uses a Cortex M7 architecture. This unit is particularly power efficient. It has a six stage superscalar pipeline and implements branch prediction. Floating point can also be supported. Internal buses are 64-bits wide.

The VA10820 microcontroller uses the Cortex M0, as does the PA32KAS and Protec products. These are available as rad-hard parts. The Mo is optimized for small size. They have a 3-stage pipeline, and are 32-bits. There is hardware multiply, but no floating point unit.

ARM 64

ARM architecture version 8 (ARMv8) defines support for 64-bit data and addressing, and multicore operations. It is dual-issue

superscalar, so almost twice as many instructions per clock can be executed. All instructions are 32-bits, allowing for more rapid decode. There are 31 general purpose registers. It can include the NEON SIMD instruction set extension, and a vector floating point unit. It supports the Jazelle architecture. Branches are optimized with an advanced branch predictor that can achieve success rates approaching 95%. The virtual address space is 48 bits, with a 40-bit physical address space supported initially. The ARM v8 ISA allows for operation in 32-bit and 64-bit modes. It includes virtualization support, NEON SIMD support, and enhanced security, while maintaining compatibility with ARMv7.

The Version 8 is implemented by Freescale Semiconductor, Samsung, TI, and others. This architecture will supplement the ARM architectures dominance in smart phones, tablets, and embedded systems, with competition in top-end desktop and server applications.

Embedded refers to special purpose computers that are a part of a larger system, as opposed to generic desktop computers, tablets, and servers. Embedded systems are for specific purposes; they are not necessarily general purpose. They may have a limited or no human interface, but usually support complex I/O. The embedded computer can be characterized by the parameters of its central processing unit, memory, and input/output (I/O). The CPU parameters of importance are speed, power consumption, and price. The memory parameters include power consumption, speed, volatility, and size or capacity. I/O characteristics must be matched to external systems components, and there are many options.

ARM Multicore

The latest ARM architecture supports multicore (currently, up to 8-core) architectures. Both symmetrical and asymmetrical implementations are included. Putting a lot of cores on a single substrate is challenging, but getting them to work together co-cooperatively and non-intrusively is difficult. The CoreLink cache

coherent interconnect system IP, for use in multicore applications, is one emerging solution. Some problems are inherently parallelizable, but most are not. Not many problem domains scale linearly with the amount of computing horsepower available. Such embarrassingly parallel applications are rarely of practical interest.

The Cortex-A9 microarchitecture comes with either a scalable multicore processor, the Cortex-A9 MPCore™ or as a more traditional processor, the Cortex-A9 single core processor. The configuration includes 16, 32, or 64KB four-way associative L1 caches, with up to 8MB of L2 cache through the optional L2 cache controller. The memory architecture of the A9 is Harvard, with separate code and data paths. It can sustain four double-word write transfers every 5 cycles. It includes a high efficiency superscalar pipeline, which removes dependencies between adjacent instructions. It has double the floating point performance of previous units. Up to a 2 GHz clock is currently feasible. Two instructions per cycle are decoded. Instruction execution is speculative, using dynamic register renaming. A similar technique is used to unroll loops in the hardware at execution time. There are four execution pipelines fed from the issue queue, and out-of-order dispatch is supported, as is out-of-order write-back. Items can be marked as non-cachable, write back, or write through.

Cache Coherency in Multicore

In multicore architectures, each CPU core may have its own L1 cache, but share L2 caches with other cores. Local data in the L1 caches must be consistent with data in other L1 caches. If one core changes a value in cache due to a write operation, that data needs to be changed in other caches as well (if they hold the same item).

This problem is well known and studies from the field of multiprocessing. The issues can be addressed by several mechanisms. In cache snooping, each cache monitors the others for

changes. If a change in value is seen, the local cached copy is invalidated. This means it will have to be re-accessed from the next level before use. A global directory of cached data can also be maintained. Several protocols for cache coherency include MSI, MESI, and others.

Raspberry Pi architectures

The Raspberry Pi is a small, inexpensive, single board computer based on the ARM architecture. It is targeted to the academic market. It originally used the Broadcom BCM2835 system-on-a-chip, which has a 700 MHz ARM processor, a video GPU, and currently 512 M of RAM. It uses an SD card for storage. The Raspberry Pi runs the GNU/linux and FreeBSD operating systems. It was first sold in February 2012. Sales reached ½ million units by the Fall. Due to the open source nature of the software, Raspberry Pi applications and drivers can be downloaded from various sites. It requires a single power supply, and dissipates a few watts. It has USB ports, and an Ethernet controller. It does not have a real-time clock, but one can easily be added. It outputs video in HDMI resolution, and supports audio output. I/O includes 8 general purpose I/O lines, UART, I2C bus, and SPI bus. The Raspberry Pi comes in 32-bit or 64-bit variants.

The Raspberry Pi design comes from the Raspberry Pi Foundation in the UK, which was formed to promote the study of Computer Science. The Raspberry Pi is seen as the successor to the original BBC Microcomputer by Acorn, which resulted in the ARM processor.

The Raspberry Pi can be used a a desktop/tablet computer for general purpose applications. The B+ version of the Pi has a Image System Pipeline (ISP) to directly handle data from a digital camera, and can be programmed with OpenGL. This is a GPU with 20 processing stages. It can handle a frame rate of 30 images per second, with 1080 pixel resolution. It uses 16-way vector code, and the pipeline operates at 250 Mhz.

Arduino Architecture

The Arduino is an open source design for a Microcontroller. There's a large variety to choose from, from simple 8-bit to 32-bit . As a microcontroller, the Arduinos are targeted to sensing and control, often real-time operations. They are not heavy duty number crunchers. They have a wide variety of I/O interfaces. If they run an operating system, it would usually be real-time. They can also just run dedicated code, without a distinct operating system. A boot loader is included.

Lacking some operating system support such as memory mapping, Arduinos are generally run without one. The code handles those operating systems functions as necessary, such as the interrupt vector setup at Power-up. Expansion modules, called shields, plug directly into the Arduino motherboard.

Inherently simpler than the Raspberry Pi, Rad-Hard Arduino-class microcontrollers are currently available.

Media Processing

Media Processing refers to operations on audio and video data structures. These are Digital Signal Processing operations. Image compression and decompression operations on streaming video in real time is an example. They can provide advanced onboard processing of sensor data, and do tasks such as pattern extraction.

The ARM NEON is a general purpose SIMD engine operating on multimedia data structures. It has a 128-bit architecture, and serves as an extension for the ARM Cortex-A. It has sixteen 128-bit registers.

The Advanced SIMD extension, marketed as NEON technology, is a combined 64- and 128-bit SIMD instruction set that provides standardized acceleration for media and signal processing applications. NEON can execute MP3 audio decoding on CPUs running at 10 MHz and can run the GSM AMR (Adaptive Multi-Rate) speech codec at no more than 13 MHz. It features a comprehensive instruction set, separate register files and independent execution hardware. NEON supports 8-, 16-, 32- and 64-bit integer and single-precision (32-bit) floating-point data and operates in SIMD operations for handling audio and video processing as well as graphics and gaming processing. In NEON, the SIMD supports up to 16 operations at the same time. The NEON hardware shares the same floating-point registers as used in VFP.

Graphics Processing

The ARM Multimedia processors are graphics processing units. They do graphics operations on graphics configured data. In the same sense that regular CPU's operate on integers and boolean data, dedicated floating point units operate on floating point data, and GPU's operate on graphics data.

A GPU can also operate on integer and floating point data, but has been optimized to address graphics data, and its operations.

Graphics data can be integer or floating point. Generally, it is organized in 1 dimensional arrays (vectors) or multi-dimensional arrays. Since we will see later that we can under use our GPU to do general purpose processing, there is nothing special about the data format. Keep in mind, the GPU does not implement logic functions.

To use the GPU in a general purpose role, we basically have to reformulate our computational problem in terms of the graphics operations the GPU provides. The OpenCL language, widely used in GPU programming, is general purpose.

A GPU is a specialized computer architecture intended to manipulate image data at high rates. The GPU devices are highly parallel, and specifically designed to handle image data, and operations on that data. They do this much faster than a programmed general purpose CPU. Most desktop machines have the GPU function on a video card or integrated with their CPU. Originally, GPU's were circuit card based. Now, they're chips, and increasingly, multicore chips. GPU operations are very memory intensive. The GPU design is customized to SIMD type operations.

The instruction set of the GPU is specific to graphics operations on block data. The requirements were driven by the demands of 2-D and 3-D video games on pc's, phones, tablets, and dedicated gaming units. As GPU units became faster and more capable, they began to consume more power (and thus generate more heat) than the associated CPU's. The GPU operations are typically memory intensive, so fast access to memory is critical.

A GPU is generally a dataflow architecture, as opposed to a control-flow, Von Neumann machine. The instructions executed depend on the inputs, to the extent that the order of execution is non-

deterministic. On general purpose machines implementing graphics processing code, the behavior would be deterministic.

Although designed to process video data, some GPU's have been used as adjunct data processors and accelerators in other areas involving vectors and matrices, with operations such as the inverse discrete cosine transform. Types of higher-level processing implemented by GPU's include texture mapping, polygon rendering, object rotation, and coordination system transformation. They also support object shading operations, data oversampling, and interpolation. GPU's find a major application area in video decoding. GPU's can be used to accelerate database operations such as gather and scatter, vertex operations. In space, these would operate on sensed data.

GPU's can tackle the embarrassingly parallel problems in engineering and physics, those that map to multiple parallel tasks that can be executed simultaneously. Examples of some of these applications include protein folding and ray tracing.

You can do general purpose computing on a GPU, although it may not be the ideal platform. It requires you to recast your computation in a way the GPU understands, which is to say, in terms of graphics. So, we might have to represent the data as a 2D or 3D object, that we can apply the GPU's operations on. GPU's are special purpose devices that have instruction sets that are not general purpose, and are intended specifically for graphics data processing, and problems that lend themselves to stream or vector processing. GPU's are stream processors, in that they operate in parallel on multiple data. Given the right problem, that is map-able into the GPU's architecture, a huge performance gain of orders of magnitude can be achieved, over regular CPU's.

One figure of merit in GPU's is their *arithmetic intensity*, defined as the number of operations per memory access. You might think of this as a computation:communication ratio.

GPU's are used for coordinate system transformations, and for science data processing. The GPU can implement such operations as shading, codec, mapping, video decoding, and 3D image manipulation.

Extensive code libraries exist for GPU's, and different problem domains, from physics modeling, to video gaming and virtual reality. API's include OpenGL and Directx. OpenGL, the Open (source) graphics libraries operate across languages and platforms. It was introduced in 1992. It is an industry standard, and claims scalability from hand-held to supercomputer. It consists of a series of library functions, callable from most computer languages. DirectX, similarly, has a set of runtime libraries. It is a Microsoft product. There are other libraries of graphics functions available as well.

GPU's need high bandwidth connections to data. They are beginning to include fast, hardware managed, multi-level caches. The architectures differ from that of general purpose caches, since the GPU is mostly accessing vector data, from consecutive memory locations. GPU's have large register files on chip to reduce access time to frequently used data.

The instruction set of the GPU is specific to graphics operations on block data. The requirements were driven by the demands of 2-D and 3-D data. As GPU units became faster and more capable, they consume more power (and thus generate more heat) than the associated CPU’s.

Single instruction, multiple data (SIMD) describes computers with multiple processing elements that perform the same operation on

multiple data simultaneously. Thus, such machines exploit data level parallelism. Vector processing is where a single operation is applied to an entire 1-dimensional array of data.

The Cortex-A15 is multi-core, and has an out-of-order superscalar pipeline. The chip was introduced in 2012, and is available from TI, NVidia, Samsung, and others. It can address a terabyte of memory. The integer pipeline is 15 stages long, and the floating point pipeline is up to 25 stages. The instruction issue is speculative. There can be 4 cores per cluster, two clusters per chip. Each core has separate 32k data and instruction caches. The level-2 cache controller supports up to 4 megabytes per cluster. DSP and NEON SIMD are supported, as is floating point. Hardware virtualization support is included.

Rad hard Arm

There are several Rad-Hard ARM microcontrollers available at this writing. There use either the M7 or M0 architecture. These are equivalent to the Arduino architectures in common use as microcontrollers.

RadHard ARM Cortex-M0

This unit is a microcontroller from Protec GmbH, a company with 30 years experience in rad-hard electronics, and a portfolio of processor and support parts. It provides a Cortex M0 cpu, operating at 50 Mhz, and using 3.3 volts. Memory includes 16k each of data and program memory. Error detection and correction is included. It can interface with up to 36 megabytes of external memory. It includes GPIO pins, that can also be used as interrupts. There are 32 general purpose counter-timers, and dual UARTS. There are dual SPI interfaces.

All internal registers are triple-modular redundant. It is hardened to a TID of 300 krad, and is latch-up immune to 100 MeV-cm^2/mg. It comes in a 1.3 x 1.3 inch, 188 pin package. Not a computational powerhouse, it is a capable controller.

SAMRH71

The SAMRH71 is a rad-hard-by-design microcontroller chip from Microchip, based on their commercial grade SAMv71. It is a 32-bit Cortex M7. The rad hard version includes Spacewire and MIL-STD-1553 I/O. The radiation performance is a LET of 62 MeV/CM^2 with an SEU greater than 20 and a TID of 100 Krad. This microcontroller operates up to 100 MHz. Besides the rad-hard part, a less-expensive radiation-tolerant part is also available. The part includes CAN and Ethernet interfaces.

VA10820

This chip, from Vorago, is based on the Cortex M0 architecture. It is 32 bits, rated beyond 300 krad, and is latch-up immune to 110 MeV-cm^2/mg. It includes 32k of data, and 128 k of program memory. For I/O, it includes dual UART's, dual I^{2}C's, and three SPI's. It comes in a 128 pin ceramic package. The memory, using EDAC, is rated at fewer than 10^{-13} errors/bit-day. It implements EDAC, and TMR on critical functions. The similar PA32KAS offers 16k data and program data

High-Performance Spaceflight Computing Program

Looming on the horizon is a product from the ongoing AFSL/NASA Next Generation Space Processor (NGSP) Project. This will not be a rad-hard Arm-based microcontroller, but rather something similar to the Raspberry Pi architecture. It will have a 64-bit product, with 8 cores. It will be a capable number cruncher,

and will support common open-source software, both operating systems and high-end applications. It is based on the ARM Cortex A53 architecture. This unit is superscalar, with dual units. It has an 8-stage pipeline, a floating point unit, hardware virtualization. and conditional branch prediction. Most instructions can be issued in pairs. It will be available as a licensable IP core.

The High Performance Spacecraft Computing Project (HPSC) is ongoing. The contract for development of the unit was let to the Boeing Company, in St. Louis. The contract value exceeds $26 million. The company is to develop the rad-hard computer based on the ARM architecture, multicore “chiplets, and associated software. Each chiplet has eight processor cores, and interfaces to memory and I/O resources. The architecture supports real-time, and parallel processing. The software environment is based on linux. Delivery of the computer will be by 2021.

Strength in numbers

As a standard ARM architecture, the NGSP will support the use of standard, open source, operating systems such as linux, and applications. One relevant open source software is NASA/GSFC's Beowulf, which allows clustering. For example, a 64-node cluster computer, using the Pi architecture has already been built. The throughput is staggering. Additions to the original software allow for load leveling across the unit. The Beowulf cluster can be built from standard off-the-shelf commercial grade parts, with one rad-hard watchdog computer keeping tabs on all the compute units. It would also do trending across the cluster, and watch for pending problems that might be avoided with a timely reset.

Such as cluster computer could be utilized to support, in situ, a distributed space system, such as a swarm or constellation of Cubesats. These would be observing the same target from different points of view, or using different sensor types. The Intelligent Constellation Executive, or “mothership” would provide local

management, control, and data processing for the swarm. This is a type of space-based sensor web, using multiple cooperating systems, managed locally.

ARM at work

There have been numerous space projects utilizing the ARM processor. To be used in a mission-critical application in space, the processor has to be insensitive to radiation damage. This involves both circuit-level and architectural (implementation) techniques for radiation hardening against both total dose and transient events, such as single event upsets (SEU's). These areas are fairly well understood, and techniques such as TMR (triple modular redundancy) and error detection and correction codes are employed. These techniques apply not only to the CPU, but also the memory and I/O circuitry as well.

Besides attitude determination and control, the onboard embedded system has a variety of housekeeping tasks to attend to.

Generally, there is a dedicated unit, sometimes referred to as the Command & Data Handler (C&DH) with interfaces with the spacecraft transmitters and receivers, the onboard data storage system, and the flight computer. The C&DH, itself a computer, is in charge of uplinked data (generally, commands), onboard data storage, and data transmission. The C&DH can forward received commands directly to various spacecraft components, or can hold them for later execution at a specified time. The C&DH has a direct connection with the science instrument(s) for that data stream. If the science instrument package has many units, there may be a separate science C&DH (SC&DH) that consolidates the sensed data, and hands it over to the C&DH for transmission to the ground. It is also common for the C&DH to hand over all commands related to science instruments to the IC&DH.

Consumables inventory

The spacecraft computer can calculate and maintains a table of consumables data, both value and usage rate. This includes available state-of-charge in the batteries, amount of thruster propellant, and any other renewable or consumable asset. This is periodically telemetered to the ground.

Thermal management

The spacecraft electronics needs to be kept within a certain temperature for proper operation. Generally, the only heat source is the Sun, and the only heat sink is deep space. There are options as to how the spacecraft can be oriented. In close orbit to a planet, the planet may also represent a heat source. Automatic thermal louvers can be used to regulate the spacecraft internal temperature. The flight computer's job is to keep the science instruments or communications antennae pointed in the right direction. This might have to be overridden in the case where the spacecraft is getting too hot or too cold.

Electrical Power/energy management

The flight computer needs to know the state-of-charge (SOC) of the batteries and whether current is flowing into or out of the batteries. It the SOC is getting too low, some operations must be suspended, and the solar panels or spacecraft itself can be re-oriented to maximize charging. In some cases, redundant equipment may be turned off, according to a predetermined load-shedding algorithm. If the spacecraft batteries are fully discharged, it is generally the end of the mission, because pointing to the Sun cannot be achieved, except by lucky accident. Spacecraft going beyond Jupiter rely on RTG's – radioisotope thermoelectric generators.

Antenna Pointing

The spacecraft communications antennae must be pointed to the

large antennae on the ground (Earth) or to a communications relay satellite in a higher orbit (for Earth or Mars). The Antennae can usually be steered in two axis, independently of the spacecraft body. This can be accomplished in the Main flight computer, or be a task for the C&DH.

Safe Hold mode

As a last resort, the spacecraft has a safe-hold or survival mode that operates without computer intervention. This usually seeks to orient the spacecraft with its solar panels to the Sun to maximize power, turn off all non-essential systems, and call for help. This can be implemented in a dedicated digital unit. It used to be the case that the safe-hold mode was implemented in analog circuitry.

Science Data Processing

I have not mentioned the science payload. Generally, the science instrument(s) have their own dedicated computers that collect the data and hand it over to the spacecraft Command & Data handling unit to be downlinked with the “housekeeping” data. Some level of science data processing can also be done onboard. As missions go further out, and collect more and more data, there is a need for some processing of the data onboard, which may take years (New Horizons is an example) to get back to Earth.

For heavy-duty on-board data crunching, we can use standard Raspberry PI's in a cluster architecture, linked with NASA's Beowulf software.

NASA Data Processing Levels Definition

0 Reconstructed, unprocessed instrument and payload data at full resolution, with any and all communications artifacts (e. g., synchronization frames, communications headers, duplicate data) removed.

1a Reconstructed, unprocessed instrument data at full resolution, time-referenced, and annotated with ancillary information, including radiometric and geometric calibration coefficients and geo-referencing parameters (e. g., platform ephemeris) computed and appended but not applied to the Level 0 data (or if applied, in a manner that level 0 is fully recoverable from level 1a data).

1b Level 1a data that have been processed to sensor units (e. g., radar backscatter cross section, brightness temperature, etc.); not all instruments have Level 1b data; level 0 data is not recoverable from level 1b data.

2 Derived geophysical variables (e. g., ocean wave height, soil moisture, ice concentration) at the same resolution and location as Level 1 source data.

3 Variables mapped on uniform spacetime grid scales, usually with some completeness and consistency (e. g., missing points interpolated, complete regions mosaic-ed together from multiple orbits, etc.).

4 Model output or results from analyses of lower level data (i. e., variables that were not measured by the instruments but instead are derived from these measurements).

The ARM processor has found extensive usage in the area of Cubesats. A Cubesat is a small, affordable satellite that can be developed and launched by college, high schools, and even individuals. The specifications were developed by Academia in 1999. The basic structure is a 10 centimeter cube, (volume of 1 liter) weighing less that 1.33 kilograms. This allows a series of these standardized packages to be launched as secondary payloads on other missions. A Cubesat dispenser has been developed, the Poly-PicoSat Orbital Deployer, that holds multiple Cubesats and dispenses them on orbit. They can also be launched from the Space

Station, via a custom airlock. ESA, the United States, and Russia provide launch services. The Cubesat origin lies with Prof. Twiggs of Stanford University and was proposed as a vehicle to support hands-on university-level space education and opportunities for low-cost space access.

Cubesats can be custom made, but there has been a major industry evolved to supply components, including space computers. It allows for an off-the-shelf implementation, in addition to the custom build. There is quite a bit of synergy between the Amsat folks and Cubesats. NASA supports the Cubesat program, holding design contests providing a free launch to worthy projects. Cubesats are being developed around the world, and several hundred have been launched.

A simple Cubesat controller can be developed from a standard embedded platform such as the Arduino. The lack of radiation hardness can be balanced by the short on-orbit lifetime. The main drivers for a Cubesat flight computer are small size, small power consumption, wide functionality, and flexibility. In addition, a wide temperature range is desirable. The architecture should support a real time operating system.

The 32-bit implementation of the Arduino architecture is a strong candidate for Cubesat onboard computers. Many implementations feature a real-time clock, which is an add-on item in the Raspberry Pi architecture. A real time clock allows for the implementation of a real-time operating system. Cubesats with Arduinos have flown in orbit. The Arduino mini on the unit from Interorbital systems incorporates a current sensor to indicate a single event upset may have occurred due to radiation. The Arduino architecture has a relatively low tolerance to radiation damage. On the International Space Station, the dual Raspberry Pi B+ based AstroPi runs student -submitted software. The units are on the ISS LAN, and can be uploaded and downloaded from the ground. They make use of WiFi and have 32G micro SD cards. Raspbian is the operating system.

Although the standard Raspberry Pi is not designed to be Rad hard, it showed a surprisingly good radiation tolerance in tests. It continued to operate through a dose of 150 krad(Si), with only the loss of USB connectivity. Several commercial cubesat flight computers are based on the ARM architecture of the Pi. When the rad-hard P emerges, the games changes significantly.

The NanoMind A712D is an onboard computer for Cubesats. It uses a 32-bit ARM cpu, with 2 megabytes of RAM, and 8 megabytes of flash memory. It can also support a MicroSD flash card. It has a Can bus and a I^2C interface. It comes with an extensive software library and real time operating system. Special applications, such as attitude determination and control code are available. It is tolerant to temperatures form -40 to 85 degrees C, but is not completely rad-hard.

The UK Space agency kicked off a project in December of 2014 called Astro Pi. It was a competition for primary and secondary schools to come up with a project and associated code for a Cubesat. Two units were taken by British Astronaut Tim Peake to the International Space Station in December of 2015. Each has a camera (one for visible spectrum, one for infrared), and each has a magnetometer as well as temperature and humidity sensors. Each unit is standard, but is housed in a purpose built aluminum case.

The PiSat is a product of the Goddard Space Flight Center. It uses a Raspberry Pi flight controller, with a battery of sensors. The case is 3D printed. The project kicked off the 2014. The software is the GSFC Core Flight System modules.

The Space environment

We'll discuss the environment in which satellites and their computers operate, assuming they survive the launch.

The space environment is hostile and non-forgiving. Is there gravity in space. Of course. It is a relationship between two masses. I the orbital case, between the satellite and the Earth. It's

just that the satellite is traveling very fast, and it balances out the gravitational pull. In fact, the satellite is in the gravity field of everything else in the solar system and universe. Most of that stuff is too far away to make much difference. But, recall, the Moon effects the oceans -we call them tides. With no gravity, no convection cooling is possible, leading to potential thermal problems. Space is a high radiation environment, being above the shielding provided by Earth's atmosphere, and the magnetic field.

There are differing environments by Mission type. For Near-Earth orbiters, there are the radiation problems of the Van Allen belts and the South Atlantic Anomaly, the thermal and vacuum environment, and the issue of atmospheric drag. This drag causes orbital decay, where the spacecraft slowly descends. There is also a drag factor from the residual atmosphere and the solar wind, and the spacecraft's orbit can be affected in other ways. All Cubesat missions are currently near Earth, although NASA is developing specialized Cubesats for planetary exploration.

Zero G issues

Zero gravity, actually, free-fall, brings with it problems. There is no convection cooling, as that relies on the different densities of warm and cool air. Any little pieces of conductive material will float around and short out critical circuitry at the worst possible time. And then, there are the strange issues.

The Hughes HS 601 series of communications spacecraft suffered a series of failures in 1992-1995 due to relays. In zero gravity, tin "whiskers" grew within the units, causing them to short. The control processors on six spacecraft were effected, with three mission failures because both primary and backup computers failed. This is now a well known materials issue, with recommendations for the proper solder to be used. In 1998, the on-orbit Galaxy IV satellite's main control computer failed due to tin whiskers.

Vacuum

The Satellites operate in vacuum. Not a perfect vacuum, but fairly close. This implies a few things. Lubricants evaporate and disappear. All materials outgas to some extent. All this stuff can find its way to condense on optical surfaces, solar arrays, and radiators.

Thermal environment

In space, things are either too hot or too cold. Cooling is by conduction to an outside surface, and then radiation to cold space. This requires heat-generating electronics to have a conductive path to a radiator. That makes board design and chip packaging complex and expensive. You get about 1 watt per square meter of sunlight in low Earth orbit. This will heat up the spacecraft, or you can convert it to electrical power with solar arrays.

Parts (and you) can be damaged by excessive heat, both ambient and self-generated. In a condition known as *thermal runaway*, an uncontrolled positive feedback situation is created, where overheating causes the part to further overheat, and fail faster.

There can be a large thermal gradient of hundreds of degrees across a satellite, where one side faces the sun, and the other side faces cold space. There is a similar situation at the planet Mercury, where one side always faces the Sun, and the other, deep space. It wiggles a little, creating what is called the "Goldilocks Zone," not too hot, not too cold.

Orbital Debris

There is a huge amount of debris in Earth orbit, including old booster rockets, failed satellites (Zombie-sats), broken solar panels, nuts and bolts, a Russian Space Suit. Space is large, but all of this stuff constitutes a hazard to ongoing missions. All this stuff is tracked and reported by the U.S. Air Force. There is a requirement now that old, end-of-life satellites have to reenter the atmosphere and burn up.

Mechanical and Structural Issues

In zero gravity, everything floats, whether you want it to or not. Floating conductive particles, bits of solder or bonding wire, can short out circuitry, internal to the chip, or on boards that were not coated.

ESD sensitivity

Solid state devices are particularly susceptible to electrostatic discharge (ESD) effects. These effects can involve very large voltages that cause device breakdown. Certain semiconductor lattice structures that have been damaged can actually "heal" over time, a process called annealing. Passive parts are sensitive to ESD as well. As parts are made smaller, the susceptibility to ESD effects increases. Proper grounding helps with ESD, providing a consistent voltage across components, without significant differences. ESD lead to sudden catastrophic failure.

Spacecraft Charging

Another problem with on-orbit spacecraft is that they are not "grounded." This can be a problem when a potential develops across the structure. Ideally, steps were taken to keep every surface linked, electrically. But, the changing phenomena has been the cause of spacecraft system failures. Where does the charge come from? Mostly, the Sun, in the forms of charged particles. This can cause surface charging, and even internal charging. Above about 90 kilometers in altitude, the spacecraft is in a plasma environment At low Earth orbit, there is a low energy but high density of the plasma. The plasma rotates with the Earth's magnetic field. The density is greater at the equator, and less at the magnetic poles. Generally, electrons with energies from 1-100 keV cause surface charging, and those over 100 keV can penetrate and cause internal charging. As modern electronics is very susceptible to electron damage, proper management of charging is needed at the design level.

Just flying along in orbit causes an electric field around the spacecraft, as any conductor traveling through a magnetic field does. If everything is at the same potential, we're good, but if

there's a difference in potential, there can be electrostatic discharge. These discharges lead to electronics damage and failure, and can also cause physical damage to surfaces, due to arcing. This has been a problem at the International Space Station.

Radiation effects

On Earth, we are shielded from most radiation by our atmosphere and magnetic field. The Sun is the major source of our radiation, and high-energy particles. Right now, based on International Space Station experience, an Astronaut can have a maximum duration in space of about a year, before receiving his/her maximum lifetime dose. At the ISS you get, in a day, what some one on the ground would accumulate in a year. Shielding is one answer, but it can be counter-productive. Sometimes, a hit by a massively energetic particle can cause a spray of lower energy particles, from the shielding itself.

A large solar flare occurred in September of 1859, and was observed by British astronomer R. C. Carrington in his private observatory on his estate outside of London. Both the associated sunspots and the flare were visible to the naked eye. The resulting geomagnetic storm was recorded by a magnetograph in Britain as well. They also recorded a perturbation in the Earth's ionosphere, that we now know is caused by ionizing x-rays. In 1859, this was all observed, but not understood. Even the ionosphere was not know to exist at the time. Now, we know a Coronal Mass Ejection from the sun, associated with a solar storm, is first seen as an energy burst hitting the Earth, and later by vast streams of charged particles, that travel slower than the speed of light. At normal levels, these particles are seen as the Northern or Southern lights. The Earth's magnetic field is affected.

What did happen, and was not immediately associated with the solar storm, was interference with the early telegraph systems of the time. The telegraph was relatively new, and wires stretched for many miles. Think of them as long antennas. The telegraph

equipment was damaged, and large arc's of electricity started fires and shocked operators. No fatalities were reported. The employees of American Telegraph Company in New York found they could transmit messages with the batteries of their systems disconnected. The Northern lights were visible from Cuba. This was the largest such solar flare in at least 500 years...and so far.

What if such a super flare occurred today? First, we would have warning from sentinel satellites such as the Solar Dynamics Observatory, that are closer to the sun, and detect the passage of particles. They can tell us about this via radio, which travels faster than the particles. So, we would have a day or so's notice. All of our modern high-technology infrastructure would be at risk of damage, from the electrical grid to the Internet. Most of our satellites would be damaged, removing services we rely on such as long distance data communication, and navigation. It would be much better to turn everything off, and ride out the storm. Even that might not prevent major damage to networks. When is the next large solar event? Even the Astrophysicists can't tell us that.

Cumulative dose and single events

The more radiation that the equipment gets, in low doses for a long time, or in high doses for a shorter time, the greater the probability of damage.

These events are caused by high energy particles, usually protons, that disrupt and damage the semiconductor lattice. The effects can be upsets (bit changes) or latch-ups (bit stuck). The damage can “heal” itself, but its often permanent. Most of the problems are caused by energetic solar protons, although galactic cosmic rays are also an issue. Solar activity varies, but is now monitored by sentinel spacecraft, and periods of intensive solar radiation and particle flux can be predicted. Although the Sun is only 8 light minutes away from Earth, the energetic particles travel much slower than light, and we have several days warning. During

periods of intense solar activity, Coronal Mass Ejection (CME) events can send massive streams of charged particles outward. These hit the Earth's magnetic field and create a bow wave. The Aurora Borealis or Northern Lights are one manifestation of incoming charged particles hitting the upper reaches of the ionosphere.

Cosmic rays, particles and electromagnetic radiation, are omni-directional, and come from extra-solar sources. Most of them, 85%, are protons with gamma rays and x-rays thrown in the mix. Energy levels range to 10^6 to 10^8 electron volts (eV). These are mostly filtered out by Earth's atmosphere. There is no such mechanism on the Moon, where they reach and interact with the surface. Solar flux energy's range to several Billion (10^9) electron volts (Gev).

The effects of radiation on silicon circuits can be mitigated by redundancy, the use of specifically radiation hardened parts, Error Detection and Correction (EDAC) circuitry, and scrubbing techniques. Hardened chips are produced on special insulating substrates such as sapphire and diamond. Bipolar technology chips can withstand radiation better than CMOS technology chips, at the cost of greatly increased power consumption. Shielding techniques are also applied. Even a small thickness of aluminum blocks many of the energetic particles. However, a problem occurs when a particle collides with the aluminum atoms, creating a cascade of lower energy particles that can also cause damage. In error detection and correction techniques, special encoding of the stored information provides a protection against flipped bits, at the cost of additional bits to store. Redundancy can also be applied at the device or box level, with the popular Triple Modular Redundancy (TMR) technique triplicating everything, and assuming the probability of a double failure is less than that of a single failure. Watchdog timers are used to reset systems unless they are themselves reset by the software. Of course, the watchdog timer circuitry is also susceptible to failure.

Spacecraft computer systems followed the trend from purpose-built custom units to those based on standard microprocessors. However, the space environment is very unforgiving in many areas, the chief one being radiation. Commercial electronic parts may not last very long in orbit.

So, we have seen control systems for spacecraft go from hardwired logic to a general purpose CPU architecture programmed with software. ASIC's, or Application Specific Integrated Circuits, are also produced in radiation hard versions. This is sort of a throw-back to the hardwired approach, but has its advantages. The next step is systems built from inherently rad-hard FPGA's or chips. This brings along a penalty in price, and speed. This is the approach that is being pursued by Boeing for NASA and the Air Force.

Rad-Hard Software

This is a concept that implements routines that check and self-check, report, and attempt to re-mediate. It is an outgrowth of the testing and self-testing of a computers' functionality, with focus on detection of radiation induced damage. We know, for example, that one of the tell-tales for radiation damage is increasing current draw. At the same time, we monitor other activities and parameters in the system. This partially addresses the problem of operating with non-radiation hardened hardware in a high radiation environment. The baseline ARM-based RaspberryPi has been radiation tested to 150 kRad, and was still operational at that point.

From formal testing results, and key engineering tools, we define likely failure modes, and possible remediation's. Besides self-test, we will have cross-checking of systems. Not everything can be tested by the software, without some additional hardware. First, we use engineering analysis that will help us define the possible hardware and software failure cases, and then define actions and remediation. This is a software FMEA, failure modes and effects analysis. None of this is new, and the approach is to collect

together best practices in the software testing area, develop a library of RHS routines, and get operational experience. Another advantage of the software approach is that we can change it after launch, as more is learned, and conditions change. If we have a cluster computer, we can do trending.

Rad Hard software runs in the background on the flight computer, and checks for the signs of pending failure from any known cause. A main cpu in the cluster monitors and trends current draw across the daughter units, and take critical action such as a reboot if it deems necessary. The Rad Hard software will keep tabs on memory by conducting continuous CRC (cyclic redundancy checks). One approach to mitigating damage to semiconductor memory is "scrubbing," where we read and write back each memory locations (being careful not to interfere with ongoing operations). This will be done by a background task that is the lowest priority in the system. Watchdog timers are also useful in getting out of a situation such as a Priority Inversion, or just a radiation-induced bit flip. There will be a pre-defined safe mode for the computer as well. Key state data from just before the fault will be stored. Unused portions of memory can be filled with defined bit patterns that can be monitored for changes. We must be certain that all of the unused interrupt vectors point to a safe area in the code, in the case of an addressing fault, and this will be automatically reloaded periodically.

Functions within the RHS include current monitoring self-diagnosis suite, spurious interrupt test, memory test(s), checksums over code, data corruption testing, memory scrub, I/O functionality test, peripherals test, stack overflow monitoring, and a watchdog timer. A complete failure modes and effects analysis will be conducted over the flight computer and associated sensors and mechanisms, and this will be used to scope and define the RHS. The systems will keep and report trending data on the flight electronics. In most cases, the only remediation is a reboot.

We can also choose to utilize a small, rad-hard recovery Arduino

controller, which is immune to radiation. The recovery computer receives heart-beat signals from the main computers and takes recovery action if they are misbehaving

Watchdog Core

For implementations in an FPGA, a separate watchdog unit can be provided in hardware. This will have the same radiation tolerance as the main computer but will be less complex, thus have a smaller cross-section for charged particles. This section can use the TMR approach, rather than the entire chip. This approach was implemented by SiFive in their

FE301 and FE540, their fourth generation flash-based FPGA chip. They are immune to radiation-induced changes to configuration.

The FPGA fabric has 150,000 logic cells, which can implement math blocks, micro (64x18) and large (1024x18) SRAM, They support dual 667 mbps DDR ports.

These parts have a flight heritage. They are manufactured in a 65 nm process. They are qualified to a MIL-STD-883, Class B spec. They support JTAG, as well as 16 Spacewire ports., and PCI Express.

The advantage of having a inherently rad-hard device that can be instantiated with the VHDL for a ARM architecture, is that you now have a rad-hard ARM. However, you have to buy an ARM architecture license first.

Radiation Damage Mitigation

Homeostasis refers to a system that monitors, corrects, and controls its own state. Our bodies do that with our blood pressure, temperature, blood sugar level, and many other parameters.

We can have the spacecraft computer monitor its own performance, or have two identical systems monitor each other. Each approach has problems. We can also choose to “triplicate” the hardware, and use external logic to see if results differ. The idea is,

two outweigh one, because the probability of a double error is less than that of a single error.

In at least one case I know of, the backup computer erroneously thought the primary made a mistake, and took over control. It was wrong, and caused a spacecraft failure.

To counter the effects of “bit flips” and other effects of radiation, the memory can be designed with error detection and correction (EDAC). Generally, this means a longer, encoded word that can detect and correct M errors. There is a trade-off with price. With EDAC memory, there is a low priority background task running on the cpu that continuously reading and writing back to memory. This process, called “memory scrubbing” will catch and correct errors.

Open Source versus Proprietary

This is a topic we need to discuss before we get too far into software. It is not a technical topic, but concerns your right to use (and/or own) software. It’s those software licenses you click to agree with, and never read. That’s what the intellectual property lawyers are betting on.

Software and software tools are available in proprietary and open source versions. *Open source software* is free and widely available, and may be incorporated into your system. It is available under license, which generally says that you can use it, but derivative products must be made available under the same license. This presents a problem if it is mixed with purchased, licensed commercial software, or a level of exclusivity is required. Major government agencies such as the Department of Defense and NASA have policies related to the use of Open Source software.

Adapting a commercial or open source operating system to a particular problem domain can be tricky. Usually, the commercial operating systems need to be used “as-is” and the source code is not available. The software can usually be configured between

well-defined limits, but there will be no visibility of the internal workings. For the open source situation, there will be a multitude of source code modules and libraries that can be configured and customized, but the process is complex. The user can also write new modules in this case.

Large corporations or government agencies sometimes have problems incorporating open source products into their projects. Open Source did not fit the model of how they have done business traditionally. They are issues and lingering doubts. NASA has created an open source license, the NASA Open Source Agreement (NOSA), to address these issues. It has released software under this license, but the Free Software Foundation has some issues with the terms of the license. The Open Source Initiative (www.opensource.org) maintains the definition of Open Source, and certifies licenses such as the NOSA.

The GNU General Public License (GPL) is the most widely used free software license. It guarantees end users the freedoms to use, study, share, copy, and modify the software. Software that ensures that these rights are retained is called free software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU project in 1989. The GPL is a copyleft license, which means that derived works can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses are the standard examples. Copyleft is in counterpoint to traditional copyright. Proprietary software “poisons” the free software, and cannot be included or integrated with it, without abandoned the GPL. The GPL cover the GNU/linux operating systems and most of the GNU/linux-based applications.

A Vendor’s software tools and Operating system or application code is usually also proprietary intellectual property. It is unusual to get the source code to examine, at least without binding legal documents and additional funds. Along with this, you get the vendor support. Open Source describes a collaborative

environment for development and testing. Use of open source code carries with it an implied responsibility to “pay back” to the community. Open Source is not necessarily free.

The Open source philosophy is sometimes at odds with the rigidtized procedures evolved to ensure software performance and reliability. Offsetting this is the increased visibility into the internals of the software packages, and control over the entire software package. Besides application code, operating systems such as GNU/linux and bsd can be open source. The programming language Python is open source, as is the popular web server Apache, and the data base MySQL.

ARM Flight software

This section discuss the software, both application code and operating systems. All spacecraft are run by on-board computers that implement the various tasks. A set of common spacecraft functions has been implements in an open-source library from NASA's Goddard Space Flight Center, Flight Software Branch. This is called the Core Flight Software. It runs under the Core Flight Executive (cFE).

The ARM architecture can host the open source Fedora linux, Debian, Gentoo, FreeBSD, NetBSD, BuildRoot (embedded). The executive is a set of mission independent reusable software services and an operating environment. Within this architecture, various mission-specific applications can be hosted. The cFE focuses on the commonality of flight software. The Core Flight System (CFS) supplies libraries and applications. Decades of flight software legacy went into the concept of the cFE. Various modules provide functions such as command and telemetry service, scheduling, limit checking,and file delivery in CCSDS format.

Linux

Linux is an open source operating system. There are actually a series of different versions of Linux. Like all operating systems, it is designed to manage computer resources – memory, programs, I/O. It handles these for desktop and server computers. The standard version does not support real-time applications. A version called RTLinux can be used, and various other non-linux or linux-like operating systems are available for real-time applications. Linux was developed and is managed by Linus Torvalds, a Finnish software engineer. Popular operating systems such as Chrome and Android are derived from Linux.

Real time operating systems still manage computer resources, but the most important thing they manage is timing. Besides Linux, BSD (Berkeley Systems Distribution version of Bell Labs Unix) is another choice. This was developed and is maintained by the University of California, Berkeley. There are many variations of BSD. Another popular real time, open source operating system is RTEMS.

cFE/cFS

The Core Flight Executive, from the Flight Software Branch, Code 582, at NASA/GSFC, is an open source operating system framework. The executive is a set of mission independent re-usable software services and an operating environment. Within this architecture, various mission-specific applications can be hosted. The cFE focuses on the commonality of flight software. The Core Flight System (cFS) supplies libraries and applications. Much flight software legacy went into the concept of the cFE. It has gotten traction within the Goddard community, and is in use on many flight projects, simulators, and test beds (FlatSats) at multiple NASA centers. The code library is managed by the Goddard Space Flight Center, and, being open source, submissions from outside sources are tested and accepted

The cFE presents a layered architecture, starting with the bootstrap

process, and including a real time operating system. At this level, a board support package is needed for the particular hardware in use. Many of these have been developed. At the OS abstraction level, a Platform support package is included. The cFE core comes next, with cFE libraries and specific mission libraries. Ap's are the 5th, or upper layer. The cFE strives to provide a platform and project independent run time environment.

The boot process involves software to get things going after power-on, and is contained in non-volatile memory. cFE has boot loaders for ARM as well as other architectures. The real time operating systems can be any of a number of different open source or proprietary products, VxWorks and RTEMS for example. This layer provides interrupt handling, a scheduler, a file system, and inter-process communication.

The Platform Support Package is an abstraction layer that allows the cFE to run a particular RTOS on a particular hardware platform. There is a PSP for desktop pc's for the cFE. The cFE Core includes a set of re-usable, mission independent services. It presents a standardized application Program Interface (API) to the programmer. A software bus architecture is provided for messaging between applications.

The Event services at the core level provides an interface to send asynchronous messages, telemetry. The cFE also provides time services.

Aps include a Health and Safety Ap with a watchdog. A housekeeping AP for messages with the ground, data storage and file manager aps, a memory checker, a stored command processor, a scheduler, a checksummer, and a memory manager. Aps can be developed and added to the library with ease.

A recent NASA/GSFC Cubesat project uses a FPGA-based system-on-a-chip architecture with Linux and the cFE. The cFE has been released into the World-Wide Open Source community, and has found many applications outside of NASA.

NASA's Software Architecture Review Board considered the cFE

in 2011. They found it a well thought-out product that definitely met a NASA need. It was also seen to have the potential of becoming a dominant flight software architectural framework. The technology was seen to be mature.

The cFS is the core flight software, a series of aps for generally useful tasks onboard the spacecraft. The cFS is a platform and project independent reusable software framework and set of reusable applications. This framework is used as the basis for the flight software for satellite data systems and instruments, but can be used on other embedded systems in general. More information on the cFS can be found at http://cFS.gsfc.nasa.gov

OSAL

The OS Abstraction Layer (OSAL) project is a small software library that isolates the embedded software from the real time operating system. The OSAL provides an Application Program Interface (API) to an abstract real time operating system. This provides a way to develop one set of embedded application code that is independent of the operating system being used. It is a form of middleware.

cFS aps

cFS aps are core Flight System (cFS) applications that are plug-in's to the Core Flight Executive (cFE) component. Some of these are discussed below.

CCSDS File Delivery (CF)

The CF application is used for transmitting and receiving files. To transfer files using CFDP, the CF application must communicate with a CFDP compliant peer. CF sends and receives file information and file-data in Protocol Data Units (PDUs) that are compliant with the CFDP standard protocol defined in the CCSDS 727.0-B-4 Blue Book. The PDUs are transferred to and from the CF application via CCSDS packets on the cFE's software bus middleware.

Limit check (LC)

The LC application monitors telemetry data points in a cFS system and compares the values against predefined threshold limits. When a threshold condition is encountered, an event message is issued and a Relative Time Sequence (RTS) command script may be initiated to respond/react to the threshold violation.

Checksum (CS)

The CS application is used for for ensuring the integrity of onboard memory. CS calculates Cyclic Redundancy Checks (CRCs) on the different memory regions and compares the CRC values with a baseline value calculated at system start up. CS has the ability to ensure the integrity of cFE applications, cFE tables, the cFE core, the onboard operating system (OS), onboard EEPROM, as well as, any memory regions ("Memory") defined by the users.

Stored Command (SC)

The SC application allows a system to be autonomously commanded 24 hours a day using sequences of commands that are loaded to SC. Each command has a time tag associated with it, permitting the command to be released for distribution at predetermined times. SC supports both Absolute Time tagged command Sequences (ATSs) as well as multiple Relative Time tagged command Sequences (RTSs).

Scheduler (SCH)

The SCH application provides a method of generating software bus messages at pre-determined timing intervals. This allows the system to operate in a Time Division Multiplexed (TDM) fashion with deterministic behavior. The TDM major frame is defined by the Major Time Synchronization Signal used by the cFE TIME Services (typically 1 Hz). The Minor Frame timing (number of slots executed within each Major Frame) is also configurable.

File Manager (FM)

The FM application provides onboard file system management

services by processing ground commands for copying, moving, and renaming files, decompressing files, creating directories, deleting files and directories, providing file and directory informational telemetry messages, and providing open file and directory listings. The FM requires use of the cFS application library.

Data Storage (DS)

The DS application is used for storing software bus messages in files. These files are generally stored on a storage device such as a solid state recorder but they could be stored on any file system. Another cFS application such as CFDP (CF) must be used in order to transfer the files created by DS from their onboard storage location to where they will be viewed and processed. DS requires use of the cFS application library.

Memory Manager (MM)

The MM application is used for the loading and dumping system memory. MM provides an operator interface to the memory manipulation functions contained in the PSP (Platform Support Package) and OSAL (Operating System Abstraction Layer) components of the cFS. MM provides the ability to load and dump memory via command parameters, as well as, from files. Supports symbolic addressing. MM requires use of the cFS application library.

Housekeeping (HK)

The HK application is used for building and sending combined telemetry messages (from individual system applications) to the software bus for routing. Combining messages is performed in order to minimize downlink telemetry bandwidth. Combined messages are also useful for organizing certain types of data packets together. HK provides the capability to generate multiple combined packets so that data can be sent at different rates.

Memory Dwell (MD)

The MD application monitors memory addresses accessed by the

CPU. This task is used for both debugging and monitoring unanticipated telemetry that had not been previously defined in the system prior to deployment. The MD application requires use of the cFS application library .

Software Bus Network (SBN)

The SBN application extends the cFE Software Bus (SB) publish/subscribe messaging service across partitions, processes, processors, and networks. The SBN is prototype code and requires a patch to the cFE Software Bus code. This is now included in the software library.

Health and Safety (HS)

The HS application provides functionality for Application Monitoring, Event Monitoring, Hardware Watchdog Servicing, Execution Counter Reporting (optional), and CPU Aliveness Indication (via UART).

Being open source, you can write your own cFS aps for specific applications, or modify existing ones. However, you should submit them back to the owner (NASA-GSFC) for review and validation so they become a part of the official package.

One particularly useful approach to storing data onboard is to use Electronic Data sheets, a standard for device and event description. For example, we might have an EDS for a battery, holding the temperature, voltage, and state of charge in a standardized format. There EDS's can then easily be held in a database, such as the open -source SQL-lite. This simplifies accessing an item in the database.

ARM Application Software

In the area of applications software and development, there is a natural synergy between Java and the ARM architecture. This is not to say that other languages such as c/c++ can’t be used. A c language to byte code compiler exists, as does those for other languages. An ARM implementation does not need to support the

ARM ISA, strangely enough. Data access is little- or big endian, with little-endian the default. The processor view of memory is that it is a linear structure of bytes, numbered in ascending order from zero. Supported data types include 8, 16, and 32-bit words. A programmer's view of the registers shows R0-R12 as general purpose, R13 as the stack pointer, R14 as the link register, and R15 as the program counter. R8 to R12 are not accessible by 16-bit instructions.

Numerous software tools and environments for the ARM exist, both proprietary and open source. These include tools from industry leaders such as Wind River (VxWorks), Green Hills, Mentor Graphics, QNX, and others. These are both development and debugging toolsets. Keil provides a software development environment for ARM, and ARM itself has a "development studio" for the chips.

Data Bases, and Electronic Data Sheets

Databases are useful for organizing data, and making it easy to access. The are many excellent database products, usually based on the Structured Query Language (SQL) model. The incoming telemetry and outgoing commands are stored, time tagged. There is no need to re-invent the wheel again here. Commercial databases (and some come in open source versions) scale well and provide the security and query features needed. SQL is the linux-based product of choice, with the SQLlite slimmed down version applicable to spacecraft onboard use.

Telemetry data, or observation data organized into Electronic Data Sheets can be stored in the onboard database. EDS's provide a structure to the characteristics and device description data.

Beowulf

The Beowulf open source clustering software was developed at NASA's Goddard Space Flight Center. It allowed the use of

commodity pc's to be used in the construction of a large parallel processor. The pc's would generally be using the linux operating system, although BSD will work. Generally, the nodes would be identical. The Message Passing Interface (MPI), and Parallel Virtual Machine software libraries are used. The original project was defined in 1998, and the units are now used world wide as an inexpensive high-end supercomputer. Beowulf clusters automatically link together at boot time, needing no particular configuration. The architecture is scalable to whatever number of computers you want, more dependent on floor space, electrical power, and cooling. 64 nodes are common.

Now, think about linking 64 Raspberry-Pi computers into a cluster. It's been done. Since the Pi's cost around $35, you have the computer architecture of a 64-node machine for around $2500. You need the networking infrastructure to link all those machines together as well, using ethernet over USB.

What is such a cluster good at? Crunching large amounts of data. With a Rad-hard Pi cluster onboard the spacecraft, large amounts of data can be examined at the source.

What type of algorithm would we run on the cluster? One example is the PNN – probabilistic neural net. This open source software is good at classification and uncovering patterns in data sets. The PNN model has four layers, the input, the pattern, the summation, and the output. PNN's have been proven to be fast and accurate. A large amount of memory space is required to store the model.

A Rad-hard Pi could serve as a watchdog over the cluster, to check for any radiation-induced effects. This would be accomplished along with each compute node running the rad-hard software, discussed later.

What Space Missions use ARM?

The Surrey Satellite Technology Nanosat Applications Platform (SNAP-1) was launched on June 28, 2000. The onboard computer (OBC) is based on Intel's StrongArm SA-1100 with 4 Mbytes of

32-bit wide EDAC protected SRAM. The error correction logic can correct 2 bits in every 8 using a modified Hamming code and the errors are "washed" from memory by software to prevent accumulation from multiple single-event upsets. There is 2 Mbytes of Flash memory containing a simple bootloader which loads the application software into SRAM.

The NanoMind A712D is an onboard computer for Cubesats. It uses as 32-bit ARM cpu, with 2 megabytes of RAM, and 8 megabytes of flash memory. It can also support a MicroSD flash card. It has a Can bus and a I^2C interface. It comes with an extensive software library and real time operating system. Special applications, such as attitude determination and control code are available. It is tolerant to temperatures form -40 to 85 degrees C, but is not completely rad-hard.

Mars Helicopter Scout

The next Mars mission in 2021 will include the 2020 Rover, which has a robotic helicopter. It will be an eye-in-the-sky, looking out for hazards, planning a path, and see things that the rover's camera can't. It will be autonomous in operation. It is a technology demonstration, planned to fly five times, during the early mission. The copter blades are a meter in diameter, and it has two counter-rotating sets. Compasses can't work on Mars due o the low magnetic field, so it will use solar tracking abd inertial guidance. It will have its own solar panels. It is carried under the rover. It is dropped to the ground, and the rover moves some distance away so it can ascend. It run linux.

Cosmos

The open source control center software (COSMOS, from Ball Aerospace) can be hosted on a Raspberry Pi class machine. A student team, headed by the author, modified it slightly to provide telemetry data from the cubesat directly an Apache web server, running on the same host machine. The telemetry data, once posted on the website, could be accessed by other computers over the

web, and by tablets and smart phones.

More interestingly, COSMOS can be hosted on a ARM architecture in a mothership, giving local control of a constellation or swarm. The one addition to the standard COSMOS product is an agent module, a virtual flight controller. This module makes decisions and takes appropriate action for the smaller units, in the scenario where the ground-based control center is not in communication. One advantage of this approach is that the controller can "learn" which actions were appropriate and effective, and modify its behavior accordingly. In a mission scenario I worked on, the swarm was on the opposite side of the Sun, from the Earth, so there were long periods of a lack of communications. This meant that the swarm had to be managed locally. This is what can be used for in-site constellation management and control.

Afterword

The game is changing. With the advent of rad-hard Arduino-class microcontrollers, and the development of a Rad-hard, Raspberry-Pi class ARM, the options for computation on smallsats has expanded. This is facilitated by the use of available, open-source software.

Glossary of Terms

1553 – Military standard data bus, serial, 1 Mbps.
1U – one unit for a Cubesat, 10 x 10 x 10 cm.
3U – three units for a Cubesat.
ACE – attitude control electronics.
A/D – analog to digital
AFRL – Air Force Research Laboratory.
ALU – arithmetic logic unit.
AMBA – (ARM) Advanced Microcontroller Bus Architecture.
ANSI – American National Standards Institute.
ANTS – Autonomous NanoTechnology Swarm..
AP – application programs.
API – application program interface; specification for software modules to communicate.
Arduino – a small, inexpensive microcontroller architecture.
ARM – originally, Acorn RISC Machine; later Advanced RISC Machine.
ASIC – Application Specific Integrated Circuit.
Baud – symbol rate; may or may not be the same as bit rate.
Beowulf – a clustering technique and software for commodity computers.
Big-endian – data format with the most significant bit or byte at the lowest address, or transmitted first.
Bootloader – initial program run after power-on or reset. Gets the computer up & going.
Bootstrap – a startup or reset process that proceeds without external intervention.
BSD – Berkeley Software Distribution version of the Bell Labs Unix operating system.
BP - bundle protocol, for dealing with errors and disconnects.
BSD – Berkeley System Distribution (of Unix). Open Source.

BSP – board support package. Customization Software and device drivers.
Buffer – a temporary holding location for data.

C – programming language from Bell Labs, circa 1972.
Cache – temporary storage between cpu and main memory.
Cache coherency – process to keep the contents of multiple caches consistent.
CAN – controller area network.
C&DH – Command & Data Handling.
cFE – Core Flight Executive – NASA GSFC reusable flight software.
CFS – NASA Core Flight Software, Open Source.
Chip – integrated circuit component.
Chiplet – architecture of the HPSC ARM module
Clock – periodic timing signal to control and synchronize operations.
Cluster – A group of similar units working together, satellites, computers, etc.
CODEC – coder/decoder.
Configware – equivalent of software for FPGA architectures; configuration information.
Constellations – a group of satellites.
COP – computer operating properly.
Coprocessor – another processor to supplement the operations of the main processor. Used for floating point, video, etc. Usually relies on the main processor for instruction fetch; and control.
Copyleft – an open source license applied to software or hardware.
Core – processor unit.
COSMOS – open source control Center and testing software, runs under linux.
CPU – central processing unit.
Cubesat – small inexpensive satellite for colleges, high schools, and individuals.
D/A digital to analog.
Database – an ordered collection of data.
D-cache – data cache.
DDR – dual data rate memory.
DMA – direct memory access.
DoD – (U.S.) Department of Defense.

DRAM – Dynamic random access memory.
DSP – digital signal processing/processor.
DTN – delay tolerant network
DUT – device under test.
ECC – error correcting code
EDAC – error detection and correction.
EDS – electronic data sheet.
Embedded system – a computer systems with limited human interfaces and performing specific tasks. Usually part of a larger system.
EMC – electromagnetic compatibility.
EMI – electromagnetic interference.
EOL – end of life.
FPGA – field programmable gate array
FPU – floating point unit
EDA – Exploratory Data Analysis.
EDAC – Error Detection and correction.
EDS – electronic data sheets.
EEPROM – Electrically Erasable programmable Memory (read-only)
FDC – fault detection and correction.
FlightLinux – NASA Research Program for Open Source code in space.
Floating point – computer numeric format for real numbers; has significant digits and an exponent.
FPGA – Field Programmable Gate Array,
FPU – floating point unit.
FRAM – ferromagnetic RAM; a non-volatile memory technology.
FTP – file transfer protocol
GEO – Geosynchronous Earth orbit.
GNC – Guidance, aviation, & Control.
GOPS – giga operations per second.
GPIO – general purpose input-output.
GNC – guidance, navigation, and control.
Gnu – recursive acronym, gnu is not unix.
GPIO – general purpose I/O.
GPL – gnu public license used for free software; referred to as the

"copyleft."
GPU - Graphics Processing Unit
GSFC – (NASA) Goddard Space Flight Center.
Handshake – co-ordination mechanism.
HDL – hardware description language.
HPCC – High Performance Computing and Communications.
HPSC – High Performance Spaceflight Computing.
Hypervisor – management of virtual domains; virtual machine monitor.
I-cache – Instruction cache.
ICD – interface control document.
IC&DH – Instrument Command & Data Handling.
IDE – Integrated Development Environment
IEEE – Institute of Electrical and Electronic engineers
IEEE-754 – standard for floating point representation and calculation.
IIC – inter-integrated circuit (I/O).
IP – intellectual property; Internet protocol.
IP core – IP describing a chip design that can be licensed to be used in an FPGA or ASIC.
IP-in-Space – Internet Protocol in Space.
ISA – instruction set architecture, the software description of the computer.
ISR – interrupt service routine, a subroutine that handles an interrupt.
I&T – integration & test.
Jazelle – (Arm) direct Java byte code execution.
JPL – (NASA) Jet Propulsion Lab of the University of California.
JVM – Java virtual machine.
Kernel – main portion of the operating system. Interface between the applications and the hardware.
Krad – kilo (10^3) rad.
Latchup – condition in which a semiconductor device is stuck in one state.
LEO – low Earth orbit.
LET – linear energy transfer.
Linux – an operating source operating system.

List – a data structure.
Little-endian – data format with the least significant bit or byte at the highest address, or transmitted last.
LRU – least recently used; an algorithm for item replacement in a cache.
MCM – MultiChip Module.
Memory leak – when a program uses memory resources but does not return them, leading to a lack of available memory.
MeV – Million (10^6) electron volts.
Microcomputer – small computer. Like what's in your phone.
Microcontroller – a cpu used with sensors and actuators, usually running real-time.
Microkernel – operating system which is not monolithic; functions execute in user space.
MIL-STD-1553 – a data communication standard for a serial bus.
MMU – memory management unit.
MPE - media processing engine.
MPI – message passing interface for cluster computers.
MPU – memory protection unit.
MRAM – Magnetorestrictive Random Access Memory.
Multicore – multiple processing cores on one substrate or chip; need not be identical.
Mutex – a software mechanism to provide mutual exclusion between tasks.
NASA – (U. S.) National Aeronautics and Space Administration.
Neon – SIMD extension for ARM
NGSP – Next Generation space Processor.
NSR – non-space rated.
NVM – non-volatile memory.
NVRAM – Nonvolatile Random access memory.
OBC – on board computer.
OBD – On-Board diagnostics.
OBP – On Board Processor.
Off-the-shelf – commercially available; not custom.
Open source – methodology for hardware or software development with free distribution and access.
OpenGL – open (source) Graphics Library.

Operating system – software that controls the allocation of resources in a computer.
OSAL – operating system abstraction layer.
Paradigm – a pattern or model
Paradigm shift – a change from one paradigm to another. Disruptive or evolutionary.
PCIE – Peripheral Component Interconnect Express (interface).
Pixel – picture element; smallest addressable element on a display or a sensor.
PNN – probabilistic neural net.
POSIX – IEEE standard operating system.
PSP – Platform Support Package.
PVM – parallel virtual machine.
Rad - unit of radiation exposure
Rad-hard – hardened to resist radiation damage.
Ram – random access memory-mapped.
RapidIO – packet switched interconnect.
Real-time – system that responds to events in a predictable, bounded time.
Register – temporary storage location for a data item.
Reset – signal and process that returns the hardware to a known, defined state.
RISC – reduced instruction set computer
RISC-v – A MIPS-based open source processor architecture.
RTOS – real time operating system.
SCP – Self-checking pairs.
SDR – software defined radio.
SDVF – Software Development and Validation Facility.
SEB – single event burnout.
SEU – single event upset.
SEL – single event latchup.
SEL – single event latchup.
SEU – single event upset (radiation induced error).
SIMD – single instruction, multiple data.
SMP – symmetric multiprocessing.
SoC – system on a chip; satellite on a chip.
SOI – silicon on insulator

SoS – silicon on sapphire – an inherently radiation-hard technology
SPW - Spacewire - high speed (160 Mbps) link.
SPI - Serial Peripheral Interface - a synchronous serial communication interface.
SQL – structured query language (for databases)
SRAM – Static Random Access Memory.
SRIO – serial rapid I/O.
SSR – solid state recorder.
Synchronous – using the same clock to coordinate operations.
System – a collection of interacting elements and relationships with a specific behavior.
System of Systems – a complex collection of systems with pooled resources.
Swarm – a collection of satellites that can operate cooperatively.
sync – synchronize, synchronized.
Thread – smallest independent set of instructions managed by a multiprocessing operating system.
Thumb (Arm) – 16 bit encoding of a subset of Arm instructions.
TID – total ionizing dose.
Thumb – 16 bit subset of ARM.
TID – total integrated dose.
TMR – triple modular redundancy
Train - – a series of satellites in the same or similar orbits, providing sequential observations.
Trust Zone – ARM security extensions.
Triplicate – using three copies (of hardware, software, messaging, power supplies, etc.). for redundancy and error control.
TRL – technology readiness level
UART – Universal Asynchronous Receiver Transmitter.
USB – universal serial bus.
VHDL – very high level design language.
Virtual memory – memory management technique using address translation.
Virtualization – creating a virtual resource from available physical resources.
VMC – vehicle management computer.

Watchdog – hardware/software function to sanity check the hardware, software, and process; applies corrective action if a fault is detected; fail-safe mechanism.

Zombie-sat – a dead satellite, in orbit.

References

Andrews, J. *Co-verification of Hardware and Software for ARM SoC Design*, 1st edition, 2004, Newnes, ISBN 9780080476902, ASIN: B001464118.

Atheshian, Peter and Zulaica, Daniel *ARM Synthesizable Design with Actel FPGAs: With Mixed-Signal SOC Applications*, 2010, McGraw-Hill Professional, ISBN 0071622810.

Badawy, Wael and Jullien, Graham *System-on-chip for Real-time Applications*. 2003, Kluwer, ISBN 1-4020-7254-6.

Barlas, Gerassimos *Multicore and GPU Programming: An Integrated Approach,* Morgan Kaufmann, 1st ed, 2014, ISBN-978-0124171374.

Challa, Obulapathi N., McNair, Janise "Distributed Data Storage on Cubesat Clusters," Advances in Computing 2013, (3) 3 pp.36-49. avail: http://article.sapub.org/10.5923.j.ac.20130303.02.html

Clark, P. E.; et al *BEES for ANTS: Space Mission Application for the Autonomous NanoTechnology Swarm,* avail: https://arc.aiaa.org/doi/abs/10.2514/6.2004-6303.

Furber, Stephen B. *ARM System-on-Chip Architectur*e (2nd Edition), 2000, Addison Wesley Professional, ISBN 9780201675191.

Furber, Stephen B. *ARM System Architecture*, 1996 Addison-Wesley, ISBN 0201403528.

Ghahroodi. Massoud M., Ozer, Emre, Bull, David SEU and SET-tolerant ARM Cortex-R4 CPU for Space and Avionics Applications, avail: www.median-project.eu/wp-content/.../median2013_submission_5.pdf

Hall, John “maddog”; Gropp, William *Beowulf Cluster Computing with Linux*, 2003, ISBN -0262692929.

Hinchey, Michael G. ; Rash, James L.; Truszkowski, Walter E.; Rouff, Christopher A., Sterritt, Roy *Autonomous and Autonomic Swarms,* avail: https://ntrs.nasa.gov/search.jsp?R=20050210015 2017-12-20T20:19:24+00:00Z

Jagger, Dave *ARM Architecture Reference Manual*, 1997, Prentice Hall, ISBN 0137362994.

Lamie, Edward *Real-Time Embedded Multithreading Using ThreadX and ARM*, Newnes; 2nd edition, 2009, ISBN 1856176010.

Lindberg, Van, *Intellectual Property and Open Source, A Practical Guide to Protecting Code*, 2008, O'Reilly Media, ISBN 0596517963.

Nicholson, J. *Starting Embedded GNU/linux Development on an ARM Architecture,* 1st ed., Newnes, Jul 2013, ISBN 9780080982366.

Patterson, David A and Hennessy, John L. *Computer Organization and Design: The Hardware/Software Interface*, ARM Edition, Morgan Kaufmann, 2011, ISBN 8131222748.

Predko, Michael *Programming and Customizing the ARM7 Microcontroller*, 2011, McGraw-Hill/Tab, ISBN 0071597573.

Schulte-Ladbeck, Dr. Regina E. *Basics of Spaceflight for Space Exploration, Space Commercialization, and Space Colonization,* 2016, ISBN-150252595X.

Seal, David ARM *Architecture Reference Manual,* 2nd Edition, 2001 Addison Wesley, ISBN 0201737191.

Sloss, Andrew; Symes, Dominic; and Wright, Chris *ARM System Developer's Guide: Designing and Optimizing System Software*, Morgan Kaufmann Series in Computer Architecture and Design, 2004, ISBN 9781558608740.

Specht, D. F. (1990). "Probabilistic neural networks".Neural Networks.3: 109–118.doi:10.1016/0893-6080(90)90049-Q.

St. Laurent, Andrew M. *Understanding Open Source and Free Software Licensing*, 2004, O'Reilly, ISBN- 0596005814.

Stakem, Patrick H. *The History of Spacecraft Computers from the V-2 to the Space Station*, 2013, PRRB Publishing, ISBN-1520216181.

Stakem, Patrick H. *Microprocessors in Space*, 2011, PRRB Publishing, ISBN-1520216343.

Stakem, Patrick H. *Embedded in Space,* 2015, PRRB Publishing, ISBN-1520215916.

Stakem, Patrick H. *Cubesat Engineering*, PRRB Publishing, 2017, ISBN-1520754019.

Stakem, Patrick H. Cubesat Constellations, Clusters, and Swarms, Stakem, PRRB Publishing, 2017, ISBN-1520767544.
Stakem, Patrick H. *Cubesat Operations*, PRRB Publishing, 2017, ISBN-152076717X.

Truszkowski, Walt, et al *Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*, Springer, 1st Edition 2009, ISBN-1846282322.

Truszkowski, Walt; Clark, P. E.;, Curtis, S.; Rilee, M. Marr, G. "ANTS: Exploring the Solar System with an Autonomous

Nanotechnology Swarm," J. Lunar and Planetary Science XXXIII (2002).

Truszkowski, Walt "Prototype Fault Isolation Expert System for Spacecraft Control," N87-29136, avail: https://ntrs.nasa.gov/search.jsp?R=19870019703,

Valvano, Jonathan W. *Embedded Systems: Introduction to the ARM Cortex-M3*, CreateSpace Independent Publishing Platform, May 26, 2012, ISBN 1477508996.

Valvano, Jonathan W. *Embedded systems: Real time Interfacing to the ARM Cortex-M3,* CreateSpace Independent Publishing Platform, 2010, ISBN-10: 1463590156.

Valvano, Jonathan W. *Embedded systems: Real-Time Operating systems for the ARM Cortex-M3,* CreateSpace Independent Publishing Platform, January 3, 2012, ISBN-10: 1466468866.

Van Someren, Alex and Atack, Carol, *ARM RISC Chip: A Programmer's Guide*, 1994 Addison Wesley, ISBN 0201624109.

Violette, Daniel P. "Arduino/Raspberry Pi: Hobbyist Hardware and Radiation Total Dose Degradation, EEE Parts for Small Missions," GSFC, 2014, avail: https://ntrs.nasa.gov/search.jsp?R=20140017620.

Yiu, Joseph *The Definitive Guide to the ARM Cortex-M0*; 2nd Edition; 2011, Newnes; ISBN 978-0123854773.

Yiu, Joseph *The Definitive Guide to the ARM Cortex-M3*, 2nd Edition, 2009, Newnes, ISBN 185617963X.

Zhang, Y. (2009). "Remote-sensing Image Classification Based on an Improved Probabilistic Neural Network". Sensors. 9 (9): 7516–7539.doi:10.3390/s90907516. PM 3290485. PMID 22400006.

Resources

- Java Virtual Machine Specification, http://java.sun.com/docs/books/vmspec/
- NASA Systems Engineering Handbook, NASA SP-2007-6105. Avail: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301.pdf
- https://www.arduino.cc/
- https://www.beowulf.org/
- Wikipedia, various

Selected documents available from www.ARM.com:

Cortex-M3 Technical Reference Manual, 2006, ARM Ltd. ARM DDI 0337E.

ARMv6-M Architecture Reference Manual, 2010, ARM Ltd. ARM DDI 0419C.

ARM Architecture Reference Manual, 2005, ARM Ltd, ARM DDI 01001.

"ARM Extends Cortex Family with First Processor Optimized for FPGA", ARM press release, March 19, 2007.

Cortex-A5 Specification Summary.

Cortex-A7 Specification Summary.

Cortex-A8 Specification Summary.

Cortex-A9 Specification Summary.

Cortex-A15 Specification Summary.

Cortex-R4 Specification Summary.

Cortex-R5 Specification Summary.

Cortex-R5 & Cortex-R7 Press Release January 31, 2011.

Cortex-R7 Specification Summary.

Cortex-M0 Specification Summary.

Cortex-M0 Instruction Set.

Cortex-M1 Specification Summary.

Cortex-M3 Specification Summary.

Cortex-M4 Specification Summary.

If you enjoyed this book, you might also be interested in some of these.

Stakem, Patrick H. *16-bit Microprocessors, History and Architecture*, 2013 PRRB Publishing, ISBN-1520210922.

Stakem, Patrick H. *4- and 8-bit Microprocessors, Architecture and History*, 2013, PRRB Publishing, ISBN-152021572X,

Stakem, Patrick H. *Apollo's Computers,* 2014, PRRB Publishing, ISBN-1520215800.

Stakem, Patrick H. *The Architecture and Applications of the ARM Microprocessors,* 2013, PRRB Publishing, ISBN-1520215843.

Stakem, Patrick H. *Earth Rovers: for Exploration and Environmental Monitoring,* 2014, PRRB Publishing, ISBN-152021586X.

Stakem, Patrick H. *Embedded Computer Systems, Volume 1, Introduction and Architecture*, 2013, PRRB Publishing, ISBN-1520215959.

Stakem, Patrick H. *The History of Spacecraft Computers from the V-2 to the Space Station*, 2013, PRRB Publishing, ISBN-1520216181.

Stakem, Patrick H. *Floating Point Computation*, 2013, PRRB Publishing, ISBN-152021619X.

Stakem, Patrick H. *Architecture of Massively Parallel Microprocessor Systems*, 2011, PRRB Publishing, ISBN-1520250061.

Stakem, Patrick H. *Multicore Computer Architecture,* 2014, PRRB Publishing, ISBN-1520241372.

Stakem, Patrick H. *Personal Robots*, 2014, PRRB Publishing, ISBN-1520216254.

Stakem, Patrick H. *RISC Microprocessors, History and Overview,* 2013, PRRB Publishing, ISBN-1520216289.

Stakem, Patrick H. *Robots and Telerobots in Space Application*s, 2011, PRRB Publishing, ISBN-1520210361.

Stakem, Patrick H. *The Saturn Rocket and the Pegasus Missions, 1965,* 2013, PRRB Publishing, ISBN-1520209916.

Stakem, Patrick H. *Visiting the NASA Centers, and Locations of Historic Rockets & Spacecraft,* 2017, PRRB Publishing, ISBN-1549651205.

Stakem, Patrick H. *Microprocessors in Space*, 2011, PRRB Publishing, ISBN-1520216343.

Stakem, Patrick H. Computer *Virtualization and the Cloud*, 2013, PRRB Publishing, ISBN-152021636X.

Stakem, Patrick H. *What's the Worst That Could Happen? Bad Assumptions, Ignorance, Failures and Screw-ups in Engineering Projects, 2014,* PRRB Publishing, ISBN-1520207166.

Stakem, Patrick H. *Computer Architecture & Programming of the Intel x86 Family, 2013,* PRRB Publishing, ISBN-1520263724.

Stakem, Patrick H. *The Hardware and Software Architecture of the Transputer*, 2011,PRRB Publishing, ISBN-152020681X.

Stakem, Patrick H. *Mainframes, Computing on Big Iron*, 2015, PRRB Publishing, ISBN- 1520216459.

Stakem, Patrick H. *Spacecraft Control Centers*, 2015, PRRB Publishing, ISBN-1520200617.

Stakem, Patrick H. *Embedded in Space,* 2015, PRRB Publishing, ISBN-1520215916.

Stakem, Patrick H. *A Practitioner's Guide to RISC Microprocessor Architecture*, Wiley-Interscience, 1996, ISBN 0471130184.

Stakem, Patrick H. *Cubesat Engineeering*, PRRB Publishing, 2017, ISBN-1520754019.

Stakem, Patrick H. *Cubesat Operations*, PRRB Publishing, 2017, ISBN-152076717X.

Stakem, Patrick H. *Interplanetary Cubesats*, PRRB Publishing, 2017, ISBN-1520766173 .

Stakem, Patrick H. Cubesat Constellations, Clusters, and Swarms, Stakem, PRRB Publishing, 2017, ISBN-1520767544.

Stakem, Patrick H. *Graphics Processing Units, an overview*, 2017, PRRB Publishing, ISBN-1520879695.

Stakem, Patrick H. *Intel Embedded and the Arduino-101, 2017,* PRRB Publishing, ISBN-1520879296.

Stakem, Patrick H. *Orbital Debris, the problem and the mitigation*, 2018, PRRB Publishing, ISBN-*1980466483.*

Stakem, Patrick H. *Manufacturing in Space*, 2018, PRRB Publishing, ISBN-1977076041.

Stakem, Patrick H. *NASA's Ships and Planes*, 2018, PRRB Publishing, ISBN-1977076823.

Stakem, Patrick H. *Space Tourism*, 2018, PRRB Publishing, ISBN-1977073506.

Stakem, Patrick H. *STEM – Data Storage and Communications*, 2018, PRRB Publishing, ISBN-1977073115.

Stakem, Patrick H. *In-Space Robotic Repair and Servicing*, 2018, PRRB Publishing, ISBN-1980478236.

Stakem, Patrick H. *Introducing Astronomy in the pre-K to 12 Curricula, A Resource Guide for Educators*, 2017, PRRB Publishing, ISBN-198104065X.
Also available in a Brazilian Portuguese edition, ISBN-1983106127.

Stakem, Patrick H. *Deep Space Gateways, the Moon and Beyond*, 2017, PRRB Publishing, ISBN-1973465701.

Stakem, Patrick H. *Exploration of the Gas Giants, Space Missions to Jupiter, Saturn, Uranus, and Neptune*, PRRB Publishing, 2018, ISBN-9781717814500.

Stakem, Patrick H. *Crewed Spacecraft*, 2017, PRRB Publishing, ISBN-1549992406.

Stakem, Patrick H. *Rocketplanes to Space*, 2017, PRRB Publishing, ISBN-1549992589.

Stakem, Patrick H. *Crewed Space Stations,* 2017, PRRB Publishing, ISBN-1549992228.

Stakem, Patrick H. *Enviro-bots for STEM: Using Robotics in the pre-K to 12 Curricula, A Resource Guide for Educators,* 2017, PRRB Publishing, ISBN-1549656619.

Stakem, Patrick H. *STEM-Sat, Using Cubesats in the pre-K to 12 Curricula, A Resource Guide for Educators*, 2017, ISBN-1549656376.

Stakem, Patrick H. *Lunar Orbital Platform-Gateway*, 2018, PRRB

Publishing, ISBN-1980498628.

Stakem, Patrick H. Embedded GPU's, 2018, PRRB Publishing, ISBN- 1980476497.

Stakem, Patrick H. Mobile Cloud Robotics, 2018, PRRB Publishing, ISBN- 1980488088

Stakem, Patrick H. *Extreme Environment Embedded Systems,* 2017, PRRB Publishing, ISBN-1520215967.

Stakem, Patrick H. *What's the Worst, Volume-2*, 2018, ISBN-1981005579.

Stakem, Patrick H., *Spaceports*, 2018, ISBN-1981022287.

Stakem, Patrick H., *Space Launch Vehicles*, 2018, ISBN-1983071773.

Stakem, Patrick H. *Mars*, 2018, ISBN-1983116902.

Stakem, Patrick H. *X-86, 40th Anniversary ed*, 2018, ISBN-1983189405.

Stakem, Patrick H. *Lunar Orbital Platform-Gateway*, 2018, PRRB Publishing, ISBN-1980498628.

Stakem, Patrick H. *Space Weather*, 2018, ISBN-1723904023.

Stakem, Patrick H. *STEM-Engineering Process*, 2017, ISBN-1983196517.

Stakem, Patrick H. *Space Telescopes,* 2018, PRRB Publishing, ISBN-1728728568.

Stakem, Patrick H. *Exoplanets*, 2018, PRRB Publishing, ISBN-9781731385055.

Stakem, Patrick H. *Planetary Defense*, 2018, PRRB Publishing, ISBN-9781731001207.

Patrick H. Stakem *Exploration of the Asteroid Belt*, 2018, PRRB Publishing, ISBN - 1731049846.

Patrick H. Stakem *Terraforming*, 2018, PRRB Publishing, ISBN-1790308100.

Patrick H. Stakem, *Martian Railroad,* 2019, PRRB Publishing, ISBN-1794488243.

Patrick H. Stakem, *Exoplanets,* 2019, PRRB Publishing, ISBN-1731385056.

Patrick H. Stakem, *Exploiting the Moon,* 2019, PRRB Publishing, ISBN-1091057850.

Patrick H. Stakem, *RISC-V, an Open Source Solution for Space Flight Computers,* 2019, PRRB Publishing, ISBN-1796434388.

Patrick H. Stakem, *Arm in Space*, 2019, PRRB Publishing, ISBN-9781099789137.

Patrick H. Stakem, *Extraterrestrial Life*, 2019, PRRB Publishing, ISBN-978-1072072188.

Patrick H. Stakem, *Space Command*, 2019, PRRB Publishing, ISBN-978-*1693005398.*